An Architecture for Defeasible-Reasoning-Based Cooperative Distributed Planning

Sergio Pajares Ferrando, Eva Onaindia, and Alejandro Torreño

Universitat Politècnica de València,
Camino de Vera, s/n
46022 Valencia, Spain
{spajares,onaindia,atorreno}@dsic.upv.es

Abstract. Cooperation plays a fundamental role in distributed planning, in which a team of distributed intelligent agents with diverse preferences, abilities and beliefs must cooperate during the planning process to achieve a set of common goals. This paper presents a **M**ulti**A**gent **P**lanning and **A**rgumentation (MAPA) architecture based on a multiagent partial order planning paradigm using argumentation for communicating agents. Agents use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge. In MAPA, actions and arguments may be proposed by different agents to enforce some goal, if their conditions are known to apply and arguments are not defeated by other arguments applying. In order to plan for these goals, agents start a stepwise dialogue consisting of exchanges of plan proposals to satisfy this open goal, and they evaluate each plan proposal according to the arguments put forward for or against it. After this, an agreement must be reached in order to select the next plan to be refined.

Keywords: Cooperative distributed planning, Defeasible Reasoning, Argumentation.

1 Introduction

A Cooperative Information System (CIS) is a large scale information system that interconnects various systems of different and autonomous organizations, geographically distributed and sharing common objectives [18]. With the emergence of new technologies in computing, such as SaaS, cloud computing, Service Oriented Computing, mash-ups, Web Services, Semantic Web, Knowledge Grid, and other approaches, it is becoming increasingly natural to deal with Agent-based computing or **MultiAgent Systems.**[28]. Agents, as distributed autonomous software entities, are required to engage in interactions, argue with one another, make agreements, and make proactive run-time decisions, individually and collectively, while responding to changing circumstances. For this reason, agents are being advocated as a next-generation model for engineering complex distributed systems.

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals of the information system. Unlike classical

R. Meersman, T. Dillon, and P. Herrero (Eds.): OTM 2011, Part I, LNCS 7044, pp. 200–217, 2011.

planning, in many real-world applications agents often have distributed contradictory information about the environment and their deductions are not always certain information, but *plausible*, since the conclusions can be withdrawn when new pieces of knowledge are posted by other agents. For this purpose, argumentation, which has recently become a very active research field in computer science [4,23], can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion.

Defeasible Logic Programming (DeLP) [9] is a framework for reasoning about defeasible information (also known as defeasible reasoning), where tentative conclusions are obtained from uncertain or incomplete information, and conclusions might no longer be valid after new information becomes available. The work in [10] (see section 3) introduces a first approach known as DeLP-POP framework, to integrate DeLP in Partial Order Planning (POP) [21], and the work in [20] (see section 3) extends DeLP-POP framework to a multiagent environment. As an example on how defeasible reasoning is introduced in these frameworks, we can view an agent as a business person who needs to travel between London and Athens, and has to build a plan to get to Athens. One may think the first action to do is to buy a flight ticket through an airline web site. However, another agent who is aware of the latest news on the Internet, might think the business man will not be able to fly due to a strike announcement in London. Under these circumstances, the second agent will put forward an argument against the first one in order to ensure that the business man accomplishes his goal to get Athens.

The motivation for introducing distributed planning in a multi-agent environment is twofold. On one hand, a multi-agent system design can be beneficial in many domains, particularly when a system is composed of multiple entities that are distributed functionally or spatially. On the other hand, distributed execution promotes the efficiency of parallel processing of actions, the robustness of the system to cope with complex planning problems and the simplicity of an incremental construction across a network of interconnected agents, thus avoiding the critical failures and resource limitations of centralized systems. In this paper, we present a **M**ulti**A**gent **P**lanning and **A**rgumentation (MAPA) architecture for cooperative distributed planning in a multiagent DeLP-POP framework, which extends and refines the preliminary work presented in [20]. This paper is organized as follows: section 2 gives a short related work; section 3 describes a background; section 4 introduces the MAPA architecture; section 5 presents the planning protocol of the architecture; and section 6 shows an example of application to validate the MAPA architecture. Finally, we conclude and present some directions for future work.

2 Related Work

This subsection is devoted to study the most relevant related works found in the literature: multi-agent argumentation, cooperative distributed planning (without defeasible reasoning) and centralized planning. Some systems that build on

argumentation apply theoretical reasoning for the generation and evaluation of arguments to build applications that deal with incomplete and contradictory information in dynamic domains. Some proposals in this line focus on planning tasks, or also called practical reasoning, i.e. reasoning about what actions are the best to be executed by an agent in a given situation. Dung's abstract system for argumentation [7] has been used for reasoning about conflicting plans and generating consistent sets of goals [2]. Further extensions of these works present an explicit separation of the belief arguments and goal arguments and include methods for comparing arguments based on the value of goals and the cost of resources [23]. The combination of defeasible reasoning and planning has been used in [22], in which the whole plan is viewed as an argument and then, defeasible reasoning about complete plans is performed. Although the work in [22] combines defeasible reasoning and partial order planning, defeasible reasoning is not used in the same way as [10]. In contrast, [10] uses arguments for warranting subgoals, and hence, defeasible reasoning is used in each step of the planning search process. In any case, none of these works apply to a multi-agent environment.

A proposal for dialogue-based centralized planning is that of [26], but no argumentation is made use of. The work in [3] presents a dialogue based on argumentation to reach agreements on plan proposals. Unlike our proposal, which focuses on an argumentative and stepwise construction of a plan, this latter work is aimed at handling the interdependencies between agents' plans. The work in [24] introduces a framework to build joint plans supported through the use of *argumentation schemes* as a mechanism of dialogue during the planning search. On the other hand, we can also find some systems that perform argumentation in multi-agent systems by using defeasible reasoning but are not particularly concerned with the task of planning [27].

3 Background

The key element of DeLP are defeasible rules (Head $-\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once other piece of knowledge is considered. Specifically, arguments (combinations of defeasible rules and facts) for conflicting pieces of information are built, and then compared to decide which one prevails. For instance, a defeasible rule like "According to Internet news, an airport strike is expected", is denoted as "$strike \multimap news$". Note that, if it occurs in London, then it will disrupt the passengers' plans for flying between London and Athens.

The principle of least commitment in Partial Order Planning makes it one of the more open planning frameworks. This is evidenced by the fact that most existing architectures for integrating planning with execution, information gathering, and scheduling are based on partial order planners. In [25], authors argue that POP-based frameworks offer a more promising approach for handling domains with durative actions, and temporal and resource constraints as compared to other planning approaches. In fact, most of the known implementations

of planning systems capable of handling temporal and durative constraints (including IxTET [12], as well as NASAÂs RAX [16]) are based on the POP paradigm. Even for simple planning tasks, partial order planners offer a higher degree of execution flexibility. In contrast, none of the known state-space planners can find parallel plans efficiently [14], and planners such as Graphplan [6] only generate a very restricted types of parallel plans. For this reason, partial order planning remains attractive when compared to state-space planning.

An extension of POP with DeLP-style argumentation, denoted DeLP-POP framework, was introduced in [10], where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments will not only be introduced to intentionally support some step of a plan, but they will also be presented to defeat or defend other supporting arguments in the plan. When actions and arguments are combined in a partial order plan, new types of interferences or threats appear [10]. These interferences need to be identified and resolved to obtain valid plans.

Finally, the work in [20] proposes a preliminary extension of the theoretical DeLP-POP framework to a multiagent environment. Specifically, it proposes a dialogue for argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. Unlike [20], the MAPA architecture presented here solves the qualification problem, identifies new types of threats, and extends the agents' knowledge bases by including a set of agent-specific preferences. This allows us to extend and adapt the planning protocol of MAPA to a fully-automated argumentative dialogue between agents so as to reach agreements during the plan construction. Moreover, the MAPA architecture promotes a more practical vision of the extension of DeLP-POP to a multi-agent environment.

4 Elements of the MAPA Architecture

In state-based planning, a plan Π is a linear sequence of actions, and thus before each action is added to the plan Π, we know which consistent state will hold. In contrast, MAPA architecture is based on POP[1], where a partial order plan Π is a set of actions whose execution ordering $\prec$ is only partially specified (thus encoding multiple linear plans).

The MAPA architecture works on a planning process distributed among several planning agents, which have an incomplete knowledge (i.e. the set of actions and arguments that an agent can propose can be different from other agents'), and have to devise a joint, non-linear plan which may be later executed by them. The following subsections expose (i) the agents' planning model and the notion of argument, (ii) the improvements introduced to deal with the qualification problem and the notion of plan, and (iii) the new definition and handling of threats introduced by the qualification problem.

[1] We consider that POP is the best planning approach concerned with the dynamic multiagent nature due to the ease to join several plan proposals into a single joint plan.

4.1 The Agents' Planning Model and Arguments

The planning model of each agent is based on a set of literals Lit, such that $\ell \in \mathsf{Lit}$ is a ground atom and $\sim\ell \in \mathsf{Lit}$ is a negated ground atom, where $\sim$represents the strong negation and $\overline{\ell} = \sim\ell$. Each agent x of the MAPA architecture is initially endowed with **a planning task** $\mathbb{M}_x = ((\Psi_x, \Delta_x), A_x, F_x, G)$ where:

1. $\Psi_x \subseteq \mathsf{Lit}$, represents a consistent set of true facts which describe the initial state of the task.
2. Δ_x is a set of defeasible rules $\delta = \ell_0, \ldots, \ell_k \prec \ell_0^{'}, \ldots, \ell_k^{'}$.
3. A_x is a set of actions $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$ where $\mathsf{P}(\alpha) \subseteq \mathsf{Lit}$ is a set of preconditions and $\mathsf{X}(\alpha) \subseteq \mathsf{Lit}$ is a set of effects.
4. F_x represents a consistent set of the agent-specific preferences $F_x \subseteq \{(a, d) \mid (a \in A),\ d \in [0, 100]\}$, where the action a is preferred with the estimated interest degree d.
5. $G \subseteq \mathsf{Lit}$ is the set of common goals which have to be satisfied.

The diversity of preferences is addressed by means of agreements between the agents during the planning process. We assume that agents are fully cooperative, so they have no incentives to retain relevant information. In POP, Ψ (consistent set of agents' initial states of the task) and G are encoded as dummy actions $\{\alpha_\Psi \prec \alpha_G\}$ with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$, and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$.

An **argument** $\mathcal{A}$ for $\ell \in \mathsf{Lit}$, is denoted as $\mathcal{A} = (\{\ell\}, \{\Delta^{'}\})$, where $\Delta^{'}$ is a subset of defeasible rules $\Delta^{'} \subseteq \Delta$. $\mathcal{A}$ is consistent if $\mathsf{base}(\mathcal{A}) \cup \mathcal{A}$ is non-contradictory.

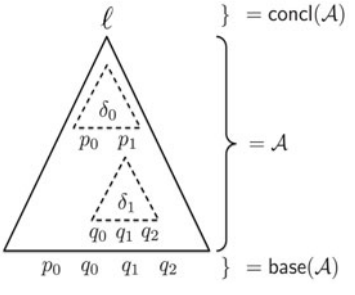

Fig. 1. An argument $\mathcal{A}$ for l using the two defeasible rules: $\delta_0 = l \prec\!\!\!\prec \{p_0, p_1\}$ and $\delta 1 = p_1 \prec\!\!\!\prec \{q_0, q_1, q_2\}$

Figure 1 shows an example of an argument proposed $\mathcal{A}$, where $\mathsf{literals}(\mathcal{A}) = \{l, p_0, p_1, q_0, q_1, q_2\}$. This argument for a literal ℓ does not suffice to warrant ℓ, it depends on the interaction among arguments (see section 5.2), which will grant consistency. Given two arguments $\mathcal{A}, \mathcal{B}$, we say $\mathcal{A}$ *attacks* $\mathcal{B}$ if the conclusion of $\mathcal{A}$ contradicts some fact used in $\mathcal{B}$, that is, if $\overline{\mathsf{concl}(\mathcal{A})} \in \mathsf{literals}(\mathcal{B})$. Therefore, the MAPA architecture semantically differentiates between supporting arguments (or **argument steps**) as the arguments specifically used to support some open

condition of the plan, and **attacking arguments** which are only introduced to attack some argument step previously introduced in the plan (i.e. it is not used to support any open condition).

4.2 The Qualification Problem and Plan Definition

The qualification problem [13], which is an important problem currently not supported in many planning architectures, is concerned with the impossibility of listing all the preconditions required for a real-world action to have its intended effect. For instance, let α (e.g. *"flying from London to Athens"*) be an action with n effects $\{e_0, e_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. e_0 =*"be at Athens city"*), which are defeated by the defeasible conditions $\{d_0, d_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. d_0 = *"Volcanic ash cloud between London to Athens"*, d_1 =*"Airport strike in London"*) respectively. Note that, if these defeasible conditions occur, the expected effects of α would not be achieved. The work in [10] solves this issue by introducing these defeasible conditions as negated preconditions of α ($\{\overline{d_0}, \overline{d_1}, \ldots\} \subseteq \mathsf{P}(\alpha)$), which must be derived by arguments.

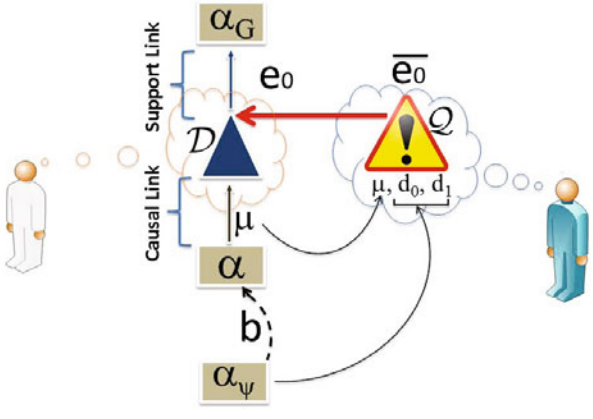

Fig. 2. An example solving the qualification problem

However, an action α in MAPA architecture follows a specific representation in order to deal with this problem. We introduce a **fictitious effect** μ (meaning α *was just executed*); then we define $\mathsf{X}(\alpha) = \{\mu\}$ and expand the set of rules Δ with $\{e_k \prec\!\!\!- \mu\} \cup \{\overline{e_k} \prec\!\!\!- \mu, d_k\}$, where e_k represents the effect of the action α and d_k is a defeasible condition. For instance, in Figure 2, the precondition e_0 of the action α_G is initially derived by an argument $\mathcal{D} = (\{e_0\}, \{e_0 \prec\!\!\!- \mu\})$ whose $\mathsf{base}(\mathcal{D}) = \mu$ will be satisfied by α. Then an attacking argument $\mathcal{Q} = (\{\overline{e_0}\}, \{\overline{e_0} \prec\!\!\!- \mu, d_0, d_1\})$, which is a defeater of $\mathcal{D}$ ($\mathcal{Q}$ attacks $\mathcal{D}$), arises from the distributed knowledge among agents. Triangles in Figure 2 represent argument steps (i.e. arguments that support preconditions of action steps), for instance the argument $\mathcal{D}$, or arguments attacking some other argument, for instance the argument $\mathcal{Q}$, and both are labeled with the argument name, while rectangles represent action

steps (i.e. actions that support the basis of an argument step) and are labeled with the action name.

The MAPA architecture defines **a plan** Π as a tuple $\Pi = (A(\Pi), Args(\Pi), G(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$, where $A(\Pi)$ denotes the set of action steps, $Args(\Pi)$ represents the set of argument steps, $G(\Pi)$ is the the task's common goals, $\mathcal{OC}(\Pi)$ is a set of ordering constraints, and $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links correspondingly. Let ℓ_1 be an open goal, motivated by some action step $\beta \in A$, i.e. $\ell_1 \in \mathsf{P}(\beta)$, and, let ℓ_2 be another open goal, motivated by some argument step $\mathcal{A} \subseteq \Delta$, i.e. $\ell_2 \in \mathsf{base}(\mathcal{A})$. Then, the goal $\ell_1 \in \mathsf{P}(\beta)$ must be supported by the argument $\mathcal{A}$, which will introduce a **support link** $(\mathcal{A}, \ell_1, \beta) \in \mathcal{SL}(\Pi)$, where $\mathcal{SL}(\Pi) \subseteq \Delta \times G(\Pi) \times A$, while the goal ℓ_2 must be satisfied by an action α, by introducing a **causal link** $(\alpha, \ell_2, \mathcal{A}) \in \mathcal{CL}(\Pi)$ where $\mathcal{CL}(\Pi) \subseteq A \times G(\Pi) \times \Delta$. Note that an argument $\mathcal{B}$ cannot support another argument $\mathcal{A}$ with a support link in $\mathcal{SL}(\Pi)$, and an action α_1 cannot support another action α_2 with a causal link in $\mathcal{CL}(\Pi)$. To get $\mathcal{B}$ to support step $\mathcal{A}$, $\mathcal{A}$ must be replaced by $\mathcal{A} \cup \mathcal{B}$, and to get α_1 to support action step α_2, an argument $(\{e_k\}, \{e_k \prec \mu\})$ must be inserted between α_1 and α_2, where $\mathsf{X}(\alpha_2) = \mu$ and $\mathsf{P}(\alpha_1) = e_k$. Additionally, unlike in DeLP-POP, ordering constraints are placed between argument steps $(\mathcal{A}, \mathcal{B}) \in \mathcal{OC}(\Pi)$, since every action (excepting α_G) is preceded by an argument which derives its actual effects.

4.3 Interferences among Actions and Arguments

If only actions are taken into account in a planning architecture, then there is only one type of destructive interference that can arise in a plan under construction. This interference is captured by the notion of threat in POP, and occurs when a new action inserted in the plan threatens (deletes) a goal solved by other action steps. When actions and arguments are combined to construct plans, new types of interferences appear that need to be identified and resolved to obtain a valid plan. In multiagent DeLP-POP [20], we identified three types of interferences or threats, that cover all the interferences that may arise in a partial plan: argument-argument, action-argument and action-action threats.

However, since the goals must be initially derived by some argument step in the MAPA architecture, and then its basis must be satisfied by another action step (including the initial step), argument-argument threats cover all the interferences that may arise in a plan dealing with the qualification problem. Nevertheless, MAPA architecture differentiates semantically between:

1. **Planning threats** (PlaThreats): Threats that arise between two argument steps. For instance, let "w" be an open condition of the plan in Figure 3(c'), then the argument with an admiration is acting as a supporting argument and a PlaThreat will be discovered. These threats override the typical action-action and action-argument threats of [20]. As we will discuss in subsection 5.1, this kind of threats will be discovered and possibly resolved (by promote or demote) in the *POP Search Tree*.

2. **Argumentation threats** (ArgThreats): Threats that arise when an agent discovers a new defeater which specifically attacks some argument step. Unlike the PlaThreats, here the attacks to some argument step are made by some attacking argument. For instance, in case that the argument with an admiration in Figure 3(c') is an attacking argument (i.e "w" is not an open goal), Figure 3(c') represents an ArgThreat. Although this kind of threat is also called argument-argument threat in [20], here we rename them to ArgThreats with the aim of distinguishing between PlaThreats and ArgThreats. As shown is subsection 5.2, these threats will be discovered and possibly resolved (by Defeat-the-defeater in Figure 3(c")) in the *POP Evaluation Tree*.

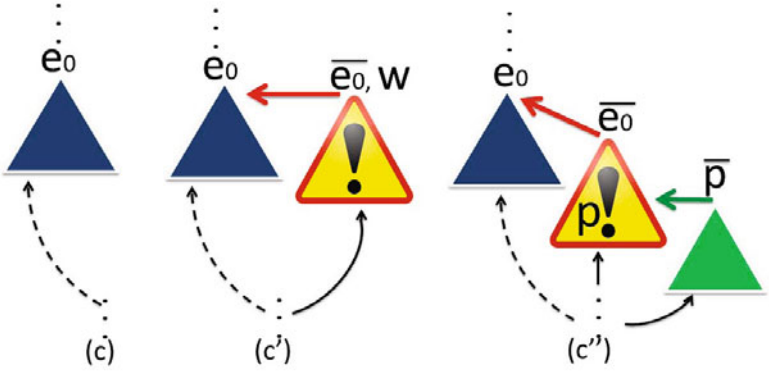

Fig. 3. (c) Selected plan. (c') Threat. (c") Solution to (c'): Defeat-the-defeater.

5 Cooperative Distributed Planning Protocol in the MAPA Architecture

Figure 4 illustrates the planning protocol, which is mainly composed of three different cooperative distributed processes among the planning agents: Plan Generation, Plan Evaluation, and Plan Selection.

Different planning heuristics such as Z-LIFO [11], or the threat detect-&-solve [10] can be used to select the next open goal to solve. In our case, we will consider turn-based dialogues, a mechanism traditionally used in cooperative scenarios where agents only participate during their turn. Additionally, agents can also be modeled to put a veto on information or decisions of other agents. Agents are enumerated, and each process is implemented through a different argumentative dialogue.

5.1 Plan Generation

The input is both the selected plan Π_r and the selected open goal (flaw) Φ, according to the Plan Selection process (see subsection 5.3) and open goal selection heuristic. The flaw Φ can be referred to both goals and PlaThreats. The main goal of this process is to allow agents to propose **a set of refinement plans**

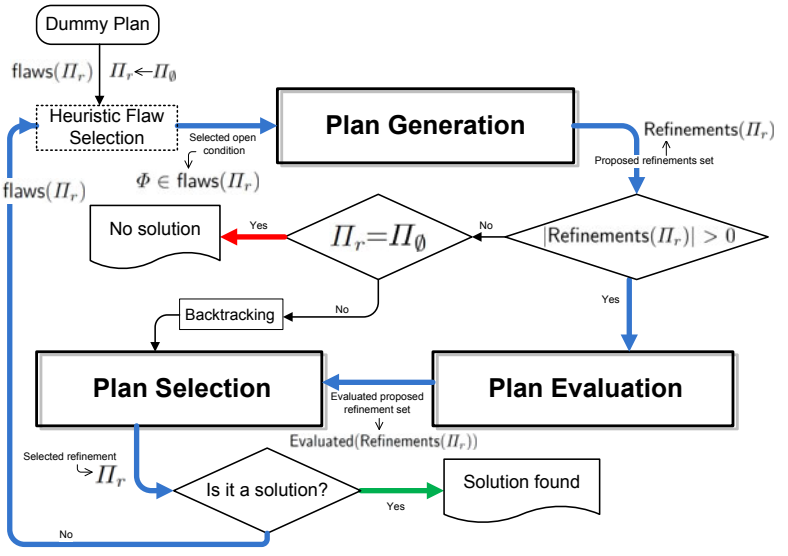

Fig. 4. Planning Protocol in the MAPA architecture

Refinements(Π_r), where each $\Pi_r(\xi) \in$ Refinements(Π_r) is a refinement step in the *POP Search Tree* that solves a selected flaw Φ such that $\Phi \in$ flaws(Π_r) and $\Phi \notin$ flaws($\Pi_r(\xi)$). Following, we explain the two steps involved in this process:

1. PROPOSALS ROUND: Each agent, at its turn, proposes alternative ways to achieve or derive Φ. The process ends when all agents have had a turn. Refinements of a plan Π_r are labeled as $\Pi_r^{(n,i)}(\xi)$, where $n \in \mathbb{Z}$ indicates the refinement proposal by the agent, $i \in \mathbb{Z}$ represents the agent, and $r \in \mathbb{Z}$ represents the selected plan by the Plan Selection process. Note that, at each turn, an agent can propose as many plans as possible from its knowledge.
2. LEARNING ROUND: Each agent updates its set of actions with the new actions which appear in the refinements proposed by other agents.

The output of this process is a set of plans Refinements(Π_r) where each $\Pi_r(\xi) \in$ Refinements(Π_r) extends Π_r. If |Refinements(Π_r)| > 0, i.e. there is at least one refinement plan, it is used as an input to the Plan Evaluation process (see section 5.2). If |Refinements(Π_r)| = 0, i.e. there is not any proposal to solve the flaw Φ, a backtracking step is performed, pruning the current base plan Π_r.

5.2 Plan Evaluation

Roughly, the problem stems from different agents discussing about a given plan; since these agents may have different initial facts and defeasible rules they may not agree on the evaluation of the plan at some step. Along with the *POP Search Tree* of the previous section, the MAPA architecture also considers the notion of *POP Evaluation Tree*.

Definition 1. *POP Evaluation Tree: Let $\Pi_r(\xi)$ be a refinement of plan Π_r from the previous process. A POP Evaluation Tree for $\Pi_r(\xi)$, denoted $\mathcal{T}_{\Pi_r(\xi)}$, where there is at least one argument step $(\mathcal{A}, \ell, \beta) \in \mathcal{SL}(\Pi_r(\xi))$, is defined as follows:*

- *The root of the tree is labeled with the plan $\langle \Pi_r(\xi) \rangle$.*
- *Each node of the first level $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)) \rangle$, is a new plan extending $\Pi_r(\xi)$ with some new defeater $\mathcal{B}$ that attacks $\mathcal{A}$, discovering a new* ArgThreat *in $\Pi_r(\xi)$.*
- *Each node of the second level $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid \xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))) \rangle$, is a new plan extending $\Pi_r(\xi, \xi^{'})$ with some new defeater $\mathcal{C}$ that attacks $\mathcal{B}$ (Defeat-the-Defeater), solving the* ArgThreat *in $\Pi_r(\xi)$.*

The input of this process is a set $\mathsf{Refinements}(\Pi_r)$ of plans proposed by the agents in the previous process. Each plan $\Pi_r(\xi) \in \mathsf{Refinements}(\Pi_r)$ represents the root of a new *POP Evaluation Tree* $\mathcal{T}_{\Pi_r(\xi)}$. Following, we explain the steps involved in this cooperative process:

1. ATTACK ROUND: It initiates an evaluation dialogue for the root plan of each $\mathcal{T}_{\Pi_r(\xi)}$, where each agent sends as many $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)) \rangle$ as they know at their turn (Figure 5). If the agent does not know how to attack a root plan, then it will skip its turn.
2. DEFENSE ROUND: It allows the agents to propose ways $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid \xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))) \rangle$ to solve discovered ArgThreats in each $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)) \rangle$. This round only applies to those *POP Evaluation Trees* which have discovered threats (Figure 5).
3. LEARNING ROUND: In this stage, each agent will update its sets of initial facts and defeasible rules, by extracting literals $\ell \in \mathsf{Lit}$ and defeasible rules, from arguments' bases and plan proposals. Unlike the previous Plan Generation process, where agents learn abilities as actions, this step is focused exclusively on the literals and defeasible rules.
4. EVALUATION: This stage marks each plan $\Pi_r(\xi) \in \mathsf{Refinements}(\Pi_r)$ as an undefeated plan, in case that defeater plans $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)) \rangle$ have not been discovered, or if they have been discovered but there is a plan $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid \xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))) \rangle$. Otherwise, $\Pi_r(\xi)$ is marked as a defeated plan.

The process ends when all the plans in $\mathsf{Refinements}(\Pi_r)$ have been evaluated. The output of this process is $\mathsf{Evaluated}(\mathsf{Refinements}(\Pi_r))$, **the set of evaluated plans** (Figure 5). As shown in the next process, undefeated plans, which constitute the most promising refinements to reach a solution, are preferred to defeated plans. However, defeated plans are kept, since each non-resolved attack could be resolved in a subsequent evaluation process.

5.3 Plan Selection

Plan selection can be done through the application of standard domain-independent heuristics for evaluating plans. These heuristics approximate the cost of a solution

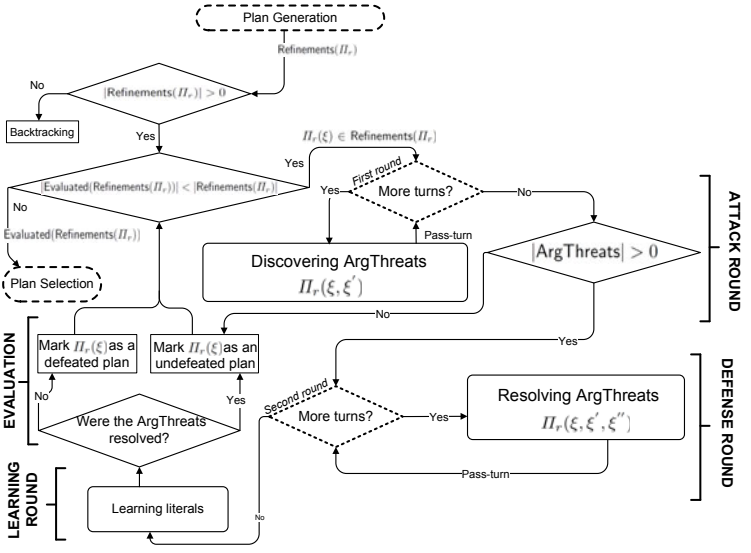

Fig. 5. Plan Evaluation protocol overview

plan in terms of the number of actions, the cost or the duration of the actions. Using this type of heuristics as the *standard rating* (R_S) for plan assessment will ignore the dynamic multi-agent nature of the MAPA architecture, where a set of preferences is assumed by each agent. Therefore, a second rating based on the agents' preferences is necessary. We will refer to it as the *preference rating* (R_F). Moreover, a third rating in terms of trust in cooperative planning is justified in [15] as a judgement about the risk attached to each component in the plan requiring cooperation, which we will call *trust rating* (R_T). R_T depends on the trust in (i) each argument step and (ii) each action step in the plan, where (i) is the trustworthiness (reputation) of the information sources which are used by the agent in order to have a perception of the environment (coded as facts and defeasible rules [10]), and, (ii) is the result of dividing the number of times the action is successfully executed into the total number of executions of the action. The MAPA architecture stores the execution of each action as a new case [1], recorded as successful if the action is executed correctly, or failure if the action failed during the execution. The success or failure of an action is determined by the achievement of the action effects. For simplicity, we only consider trust in action steps.

Unlike the Plan Generation and the Plan Evaluation process, where agents reason about agent facts, defeasible rules and actions, here agents reason about standard ratings, preference ratings, and trust ratings, considering a compromise between the desire to minimize the computational overhead and that of maximizing the quality of the plan. This process receives as input the set of evaluated plans Evaluated(Refinements(Π_r)) from the Plan Evaluation process and a set of previously not-selected partial plans OtherRefinements, in order to select a new plan $\Pi_r \in \{\text{Evaluated}(\text{Refinements}(\Pi_r)) \bigcup \text{OtherRefinements}\}$ as output. Following, we explain the steps involved in this process:

1. PLAN FILTERING: The aim is to guide the plan search by selecting the best subset $\mathsf{FilteredPlans} \subseteq \{\mathsf{Evaluated}(\mathsf{Refinements}(\Pi_r)) \bigcup \mathsf{OtherRefinements}\}$ (as candidate plans), according to the highest $R_S(\Pi)$ and $R_T(\Pi)$, such that $\Pi \in \{\mathsf{Evaluated}(\mathsf{Refinements}(\Pi_r)) \bigcup \mathsf{OtherRefinements}\}$[2], where:
   - $R_S(\Pi) = (\mathsf{cost}(\Pi) + \mathsf{heuristic}(\Pi))$, where $\mathsf{heuristic}(\Pi)$ is a heuristic estimation of the cost of reaching a solution plan $\Pi^\star$ from Π, and,
   - $R_T(\Pi)$ is the product of the trust values of the action steps in Π.
2. PLAN RANKING: The agents ($i \in \{1, 2, \ldots, k\}$) calculate their preference ratio for each candidate plan $\Pi_n \in \mathsf{FilteredPlans}$. For this purpose, they review whether each action $a \in A(\Pi_n)$ is preferred by them. If an action $a_1 \in A(\Pi_n)$ is preferred ($(a_1, d_1) \in F_i \mid d_1 > 50$), then they increase by one the value $R^i_F(\Pi_n)$; if they do not prefer an action $a_2 \in A(\Pi_n)$, ($(a_2, d_2) \in F_i \mid d_2 <= 50$), then they subtract one unit from $R^i_F(\Pi_n)$, and otherwise they keep $R^i_F(\Pi_n)$ unchanged. This stage, which simulates a internal reasoning process for or against to select each plan Π_n, allows each agent to establish a preference relation between the plans in $\mathsf{FilteredPlans}$.
3. PLAN NEGOTIATION: Since each agent has identified its preferred candidate plans, now the purpose of the negotiation is to reach an **agreement** about the next base plan $\Pi_r \in \mathsf{FilteredPlans}$. This stage can range from a simple voting process to a more sophisticated negotiation mechanism.

Finally, $\{\mathsf{NonSelectedPlans} \subset \mathsf{Evaluated}(\mathsf{Refinements}(\Pi_r)) \mid \Pi_r \notin \mathsf{NonSelectedPlans}\}$ is added to the set $\mathsf{OtherRefinements}$, and the process returns the agreed plan Π_r. If Π_r is not a solution, the control will be passed to the Heuristic Flaw Selection (see Figure 4). Otherwise, the planning process will end successfully.

6 Evaluating the MAPA Architecture within the Context of a Transit Journey Planning Service

Transit users generally know their origin and destination cities. Based on the schedules provided by the transit agencies, users choose the best routes that match their travel needs. For this purpose, a Transit Journey Planning Service (TJPS) (a specialized electronic search engine) is used to find the best route between two locations by using some means of transportation. TJPSs are being widely used by transit agencies accessed through a web user interface on a computer terminal to support clients' requests on public transport information. Most of the existing TJPSs, provided by transit agencies and companies (Google Transit Planner, Transport Direct, Transport for London, Trip Planning Tool etc.)[3], are based on static schedule data. To the best of our knowledge, these centralized planners (i) do not react to environmental changes such as bad weather, traffic jams or bad railroads, and therefore they do not provide support to defeasible

[2] Undefeated plans are preferred over defeated plans.

[3] `http://www.google.com/transit`, `http://www.transportdirect.info`, `http://www.journeyplanner.org`, `http://www.networkedtraveler.org`

reasoning, and (ii) are not able to work in a cooperative distributed environment so there is no choice for exchanging information between them.

However, defeasible reasoning is becoming an increasingly important feature in many environments where context awareness in not fully specified. The work in [5] presents a potential application for distributed defeasible reasoning in ambient computing environments, where the ambient agents, who have different viewpoints, have to face the available context. Similarly, defeasible reasoning is also being applied to semantic web and e-commerce [17]. Here, we present a novel application of cooperative distributed defeasible planning to a TJPS problem.

The MAPA architecture is implemented in Magentix 2[4], a platform for open Multiagent Systems based on the Apache Qpid[5] implementation of AMQP [6] for communication between agents. This platform incorporates a security module which provides key features regarding security, privacy, openness and interoperability not offered by other current agent platforms.

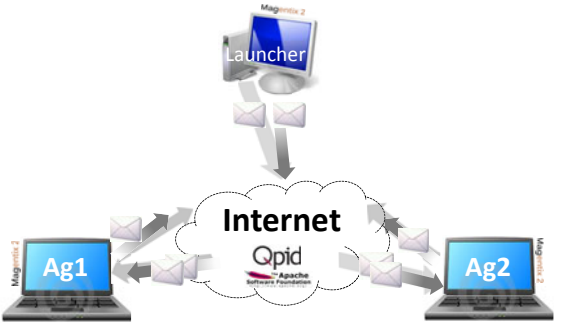

Fig. 6. Deploying the MAPA architecture

6.1 Preliminaries

According to the multi-agent systems paradigm, we have implemented a CIS as a collection of software agents (Figure 6) in the MAPA architecture. Each agent simulates an information system, and interacts with the others so as to achieve the common goals, thus forming a multi-agent society. More specifically, we consider a scenario with six different cities and two geographically distributed transit agencies, $Ag1$ (Greece transit agency) and $Ag2$ (UK transit agency), aimed at providing a customer with a plan to travel from *London* to *Athens* (Figure 7). The agencies are implemented as agents and have different knowledge (knowledge is fully distributed), so two pieces of information derived from each agent may appear to be contradictory. There are several ways to travel between both cities: via car, ship, train or plane. Let's assume that $Ag1$ uses *BBC News* as a source of information, but $Ag2$ prefers *CNN News* to keep up to date, and

[4] http://www.gti-ia.dsic.upv.es/sma/tools/magentix2/index.php
[5] http://qpid.apache.org/
[6] http://www.amqp.org

both agree on finding a plan that minimizes the journey duration. The planning tasks of the agents are defined in Figure 8, where we consider propositional STRIPS [8] planning representation.

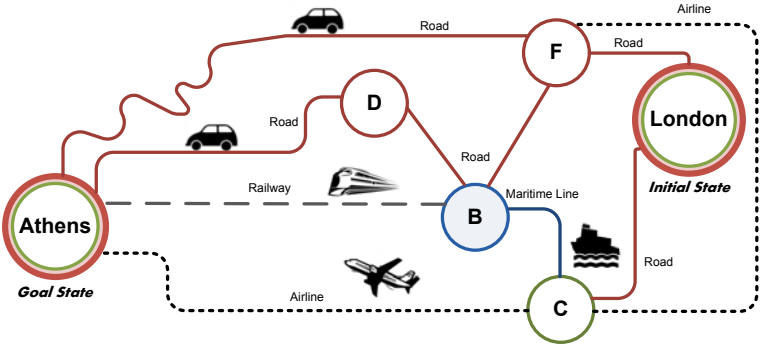

Fig. 7. Scenario of the application example

In what follows, we define the meaning of each literal and action. Literals:

- A, L - Athens, London; B, C, D, F - Other cities,
- cus, *car*, *tra*, *pl*, *shi* - a customer, a car, a train, a plane, a ship,
- *r*, *rl*, *al*, *ml* - a road, a railway, an airline company, a maritime line,
- *bw*, *sn*, *wg*, *va*, *ds*, *aeo* - bad weather, snow, wind gusts, volcano ash cloud, dangerous situation, airplane engines work well (after test),
- *br*, *ll*, *esf*, *fp* - bad railroad, landslides, electrical supply failure, flying panic,
- *h*, *tj*, *kudBBC*, *kudCNN* - holidays, traffic jam, kept up to date by BBC news, kept up to date by CNN news, and,
- μ_C, μ_P, μ_T, μ_S - moved car, moved plane, moved train and moved ship.

Actions are the following (notation: $\mathsf{X}(\alpha) \overset{\alpha}{\leftarrow} \mathsf{P}(\alpha)$, i.e. the action effects are indicated on the left side, while the action preconditions on the right side):

1. $mP(pl, x, y)$: moving plane 'pl' from location 'x' to 'y' takes 2 time units and 400 cost units.
2. $mT(tra, x, y)$: moving train 'tra' from location 'x' to 'y' takes 6 time units and 200 cost units.
3. $mS(shi, x, y)$: moving ship 'shi' from location 'x' to 'y' takes 3 time units and 100 cost units.
4. $fMc(car, x, y)$: fast-moving car 'car' from location 'x' to 'y' takes 8 time units and 80 cost units.

6.2 Implementation

The planning process starts with an empty plan $\Pi_\emptyset = \{\alpha_\Psi \prec \alpha_G\}$ and $\mathsf{flaws}(\Pi_\emptyset) = \{(at\ \mathsf{cus}\ A)\}$. First, the MAPA architecture enters the **Plan Generation** process, where four plans are suggested: i) taking the car between D and A, $\Pi_\emptyset^{(1,Ag1)}(\xi)$,

$$\Psi_{Ag1} = \{\ (wg\ B\ A);\ aeo;\ kudBBC; (at\ \mathsf{cus}\ L); fp;\ (at\ pl\ C);\ (at\ car\ L);\ (link\ al\ C\ A);\ (link\ r\ D\ A); \dots\}$$

$$\Psi_{Ag2} = \{\ kudCNN;\ fp;\ (at\ \mathsf{cus}\ L);\ (at\ tra\ B);\ (at\ shi\ C)\ (link\ rl\ B\ A);\ (link\ ml\ C\ B);\ (link\ r\ F\ A); \dots\}$$

$$\Delta_{Ag1} = \left\{ \begin{array}{c} \{(at\ pl\ ?y), (at\ \mathsf{cus}\ ?y)\} \prec (\mu_P\ ?x\ ?y);\ \{\sim(at\ tra\ ?y), \sim(at\ \mathsf{cus}\ ?y)\} \prec \{(\mu_T\ ?x\ ?y), (br\ ?x\ ?y)\}; \\ \{(at\ car\ ?y), (at\ \mathsf{cus}\ ?y)\} \prec (\mu_C\ ?x\ ?y);\ \{\sim(at\ shi\ ?y), \sim(at\ \mathsf{cus}\ ?y)\} \prec \{(\mu_S\ ?x\ ?y), (ss\ ?x\ ?y)\}; \\ (br\ ?x\ ?y) \prec (esf\ ?x\ ?y);\ (esf\ ?x\ ?y) \prec (sn\ ?x\ ?y);\ (sn\ B\ A) \prec kudBBC;\ \sim(va\ C\ A) \prec aeo; \dots \end{array} \right\}$$

$$\Delta_{Ag2} = \left\{ \begin{array}{c} \{\sim(at\ pl\ ?y), \sim(at\ \mathsf{cus}\ ?y)\} \prec \{(\mu_P\ ?x\ ?y), (ds\ ?x\ ?y)\};\ \{(at\ tra\ ?y), (at\ \mathsf{cus}\ ?y)\} \prec (\mu_T\ ?x\ ?y); \\ \{\sim(at\ car\ ?y), \sim(at\ \mathsf{cus}\ ?y)\} \prec \{(\mu_C\ ?x\ ?y), (tj\ ?x\ ?y)\};\ \{(at\ shi\ ?y), (at\ \mathsf{cus}\ ?y)\} \prec (\mu_S\ ?x\ ?y); \\ (ds\ ?x\ ?y) \prec (va\ ?x\ ?y);\ (va\ C\ A) \prec kudCNN;\ \sim(ll\ ?x\ ?y) \prec \sim(bw\ ?x\ ?y); \\ \sim(ll\ ?x\ ?y) \prec \sim(bw\ ?x\ ?y);\ \sim(bw\ B\ A) \prec kudCNN;\ \sim(sn\ B\ A) \prec kudCNN; \\ (tj\ ?x\ ?y) \prec \{(h\ ?x)(link\ r\ ?x\ ?y)\};\ (h\ F) \prec kudCNN;\ \dots \end{array} \right\}$$

$$A_{Ag1} = \left\{ \begin{array}{l} 1.\ (\mu_C\ ?x\ ?y) \xleftarrow{fMc} \{(link\ r\ ?x\ ?y),\ (at\ car\ ?x),\ (at\ \mathsf{cus}\ ?x)\} \\ 2.\ (\mu_P\ ?x\ ?y) \xleftarrow{mP} \{(link\ al\ ?x\ ?y),\ (at\ pl\ ?x),\ (at\ \mathsf{cus}\ ?x)\} \end{array} \right\}$$

$$A_{Ag2} = \left\{ \begin{array}{l} 3.\ (\mu_T\ ?x\ ?y) \xleftarrow{mT} \{(link\ rl\ ?x\ ?y),\ (at\ tra\ ?x),\ (at\ \mathsf{cus}\ ?x)\} \\ 4.\ (\mu_S\ ?x\ ?y) \xleftarrow{mS} \{(link\ ml\ ?x\ ?y),\ (at\ shi\ ?x),\ (at\ \mathsf{cus}\ ?x)\} \end{array} \right\}$$

$$F_{Ag2} = \{\ (mT,\ 90);\ (mP\ 0);\ (mS\ 60)\ \}$$

$$F_{Ag1} = \{\ (fmC\ 70);\ (mT,\ 80);\ (mP\ 5)\ \}$$

$$G = \{(at\ \mathsf{cus}\ A)\}$$

Fig. 8. Initial facts, defeasible rules, actions, preferences and common goals

or ii) between F and A, $\Pi_{\emptyset}^{(2,Ag1)}(\xi)$, iii) taking the train between B and A, $\Pi_{\emptyset}^{(1,Ag2)}(\xi)$, and iv) taking the plane between C and A, $\Pi_{\emptyset}^{(3,Ag1)}$. We only show the plan iv) $\Pi_{\emptyset}^{(3,Ag1)}(\xi) = \{(mP,\ (\mu_P\ C\ A), \mathcal{A}^{\mathrm{Ag1}}), (\mathcal{A}^{\mathrm{Ag1}}, (at\ \mathsf{cus}\ A), \alpha_G)\}$ where $\mathcal{A}^{\mathrm{Ag1}} = (\{(at\ \mathsf{cus}\ A)\}, \{(at\ \mathsf{cus}\ A) \prec (\mu_P\ C\ A)\})$ (see Figure 10(a)). The agents learn the actions they did not know from these plans.

Second, the **Plan Evaluation** process starts, where: i) $\Pi_{\emptyset}^{(1,Ag1)}(\xi)$ is not attacked by any defeater and it is labeled as an undefeated plan. ii) $\Pi_{\emptyset}^{(2,Ag1)}(\xi)$ is attacked because city F is on holiday, so a traffic jam can be expected in the road between F and A, and, therefore, the effects of the action fMc may not be satisfied. Since there are not proposals to solve this $\mathsf{ArgThreat}$, $\Pi_{\emptyset}^{(2,Ag1)}(\xi)$ is labeled as a defeated plan. iii) $\Pi_{\emptyset}^{(1,Ag2)}(\xi)$ receives one attack because snow is expected between B and A, so an electrical failure that damages the railroad between B and A might occur. If this happens, the effects of the action mT may not be satisfied. Here, $Ag2$ proposes a Defeat-the-defeater, which justifies that snow conditions are not expected between B and A, and then $\Pi_{\emptyset}^{(1,Ag2)}(\xi)$ is labeled as undefeated plan. iv) $Ag2$ attacks $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$ with $\langle \Pi_{\emptyset}^{(3,Ag1)}(\xi, \xi') \mid \xi' = \mathsf{Defeater}(\mathcal{B}^{\mathrm{Ag2}}, (\mathcal{A}^{\mathrm{Ag1}}, (at\ \mathsf{cus}\ A), \alpha_G))\rangle$ where $\mathcal{B}^{\mathrm{Ag2}} = (\{\sim(at\ \mathsf{cus}\ A)\}, \{\sim(at\ \mathsf{cus}\ A) \prec \{(\mu_P\ C\ A), (ds\ C\ A)\}; (ds\ C\ A) \prec (va\ C\ A);\ (va\ C\ A) \prec kudCNN\})$ (see Figure 9) because the volcano ashes are expected between the city C and A according to the *CNN News*, but $Ag1$ moves against $(ds\ C\ A)$ with $\langle \Pi_{\emptyset}^{(3,Ag1)}(\xi, \xi', \xi'') \mid \xi'' = \mathsf{Defeater}(\mathcal{C}^{\mathrm{Ag1}}, \mathsf{Defeater}(\mathcal{B}^{\mathrm{Ag2}}, (\mathcal{A}^{\mathrm{Ag1}}, (at\ \mathsf{cus}\ A), \alpha_G)))\rangle$ where $\mathcal{C}^{\mathrm{Ag1}} =$

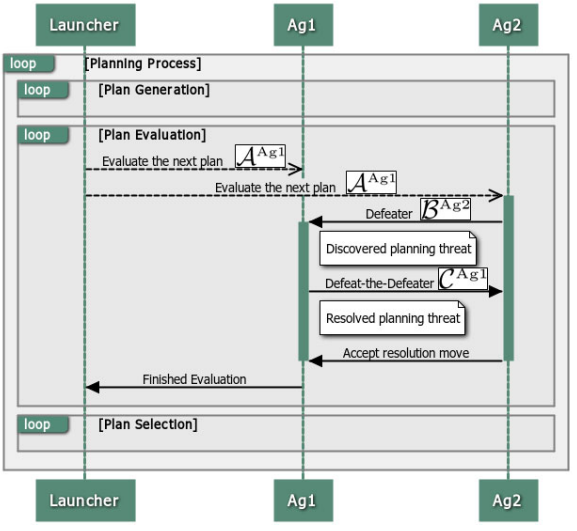

Fig. 9. Discussing about the plan $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$ in the Plan Evaluation process

$(\{\sim(va\ C\ A)\}, \{\sim(va\ C\ A) \prec aeo\})$ (see Figure 9). It is a Defeat-the-defeater resolution move since $\sim \mathsf{concl}(\mathcal{C}^{\mathrm{Ag1}}) \in \mathsf{literals}(\mathcal{B}^{\mathrm{Ag2}})$) (see Figure 10(a")), and then $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$ is labeled as an undefeated plan. The agents learn the literals and defeasible rules, they do not know at the beginning of the turn.

Third, the **Plan Selection** process starts. The best subset of plans is defined as $\mathsf{FilteredPlans} = \{\Pi_{\emptyset}^{(1,Ag2)}(\xi),\ \Pi_{\emptyset}^{(3,Ag1)}(\xi)\}$, since $\Pi_{\emptyset}^{(2,Ag1)}(\xi)$ was labeled as a defeated plan and $\mathsf{heuristic}(\Pi_{\emptyset}^{(1,Ag1}(\xi))$ returns a high value. Finally, agents

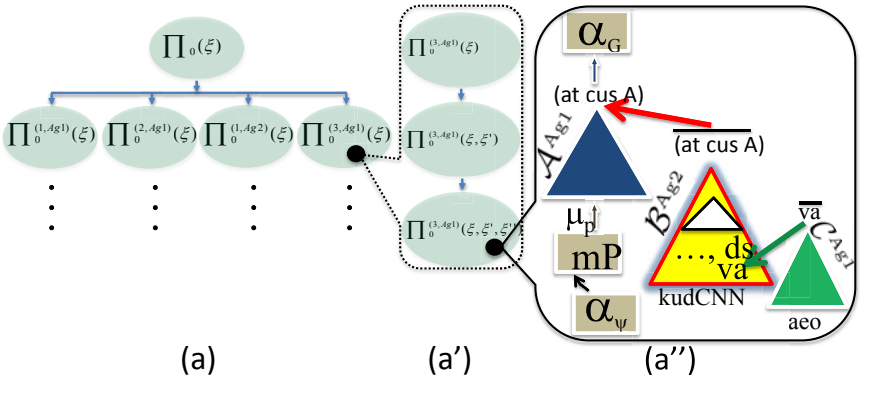

Fig. 10. Screenshots: (a) The *POP Search Tree*. (a') The *POP Evaluation Tree* for the plan $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$. (a") Viewing the content of the plan $\Pi_{\emptyset}^{(3,Ag1)}(\xi, \xi', \xi'')$.

choose $\Pi_r = \Pi_{\emptyset}^{(1,Ag2)}(\xi)$ as they prefer to take the train because of their fear of flying. For space reasons, we omit the rest of the plan search.

7 Conclusions and Future Work

We have presented MAPA, a decentralized architecture for cooperative planning in multiagent DeLP-POP, dealing with the qualification problem. It is implemented as three independent cooperation processes between agents of a team who propose, criticize, defend and select alternative plans by means of arguments and actions. For future work, we intend to work in several directions: extending MAPA to other multi-agent scenarios like argumentation-based negotiation or to temporal planning [19]; and evaluating MAPA in applications of dynamic networked cooperative business processes and knowledge-sharing, including the ability to work with and within complex supply chains. Finally, evaluating the efficiency and effectiveness of the MAPA architecture.

Acknowledgments. This work is mainly supported by FPU grant reference AP2009-1896 awarded to Sergio Pajares Ferrando and projects TIN2008-06701-C03-01, Consolider Ingenio 2010 CSD2007-00022, and PROMETEO/2008/051.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
2. Amgoud, L.: A formal framework for handling conflicting desires. In: Nielsen, T.D., Zhang, N.L. (eds.) ECSQARU 2003. LNCS (LNAI), vol. 2711, pp. 552–563. Springer, Heidelberg (2003)
3. Belesiotis, A., Rovatsos, M., Rahwan, I.: Agreeing on plans through iterated disputes. In: 9th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2010), pp. 765–772 (2010)
4. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence 171(10-15), 619–641 (2007)
5. Bikakis, A., Antoniou, G.: Distributed defeasible contextual reasoning in ambient computing. In: Aarts, E., Crowley, J.L., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.) AmI 2008. LNCS, vol. 5355, pp. 308–325. Springer, Heidelberg (2008)
6. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artificial Intelligence 90(1-2), 281–300 (1997)
7. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–358 (1995)
8. Fikes, R.E., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2(3-4), 189–208 (1971)
9. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming 4(2), 95–138 (2004)
10. García, D.R., García, A.J., Simari, G.R.: Defeasible reasoning and partial order planning. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 311–328. Springer, Heidelberg (2008)

11. Gerevini, A., Schubert, L.: Accelerating partial-order planners: Some techniques for effective search control and pruning. Journal of Artificial Intelligence Research 5, 95–137 (1996)
12. Ghallab, M., Laruelle, H.: Representation and control in ixtet, a temporal planner. In: 2nd International Conferende on Artificial Intelligence Planning Systems (AIPS 1994), vol. 94, pp. 61–67 (1994)
13. Ginsberg, M.L., Smith, D.E.: Reasoning about action II: The qualification problem. Artificial Intelligence 35(3), 311–342 (1988)
14. Haslum, P., Geffner, H.: Admissible heuristics for optimal planning. In: 5th International Conferende on Artificial Intelligence Planning Systems (AIPS 2000), pp. 140–149 (2000)
15. Ibbott, C.J., O'Keefe, R.M.: Trust, planning and benefits in a global interorganizational system. Information Systems Journal 14(2), 131–152 (2004)
16. Jonsson, A., Morris, P., Muscettola, N., Rajan, K., Smith, B.: Planning in interplanetary space: Theory and practice. In: 5th International Conference on Artificial Intelligence Planning and Scheduling (ICAPS 2000), pp. 177–186 (2000)
17. Kontopoulos, E., Bassiliades, N., Antoniou, G.: Visualizing semantic web proofs of defeasible logic in the dr-device system. Knowledge-Based Systems 24(3), 406–419 (2011)
18. Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., Batini, C.: Managing data quality in cooperative information systems. In: Chung, S., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 486–502. Springer, Heidelberg (2002)
19. Pajares, S., Onaindía, E.: Temporal defeasible argumentation in multi-agent planning. In: 22nd International Joint Conferences on Artificial Intelligence (IJCAI 2011), pp. 2834–2835 (2011)
20. Pardo, P., Pajares, S., Onaindía, E., Godo, L., Dellunde, P.: Multiagent argumentation for cooperative planning in delp-pop. In: 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), pp. 971–978 (2011)
21. Penberthy, J.S., Weld, D.: Ucpop: A sound, complete, partial order planner for adl. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning (KR 1992), pp. 103–114 (1992)
22. Pollock, J.: Defeasible planning. In: Proceedings of the AAAI Workshop, Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments. Carnegie Mellon University (June 1998)
23. Rahwan, I., Amgoud, L.: An argumentation-based approach for practical reasoning. In: International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), pp. 74–90 (2007)
24. Sapena, O., Torreño, A., Onaindía, E.: On the construction of joint plans through argumentation schemes. In: 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), pp. 1195–1196 (2011)
25. Smith, D.E., Frank, J., Jónsson, A.K.: Bridging the gap between planning and scheduling. The Knowledge Engineering Review 15(1), 47–83 (2000)
26. Tang, Y., Norman, T., Parsons, S.: A model for integrating dialogue and the execution of joint plans. In: International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), pp. 60–78 (2010)
27. Thimm, M.: Realizing argumentation in multi-agent systems using defeasible logic programming. In: International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), pp. 175–194 (2009)
28. Wooldridge, M.: Agent-based software engineering, vol. 144, pp. 26–37 (1997)